PATENT
Attorney Docket No: 200310172-1
Express Mail Label No: EV 210655533 US

SYSTEM AND METHOD FOR PROCESSING CONFIGURATION INFORMATION

RELATED APPLICATIONS

[0001] This application is related to the following commonly owned and co-filed U.S. Patent applications, filed January 30, 2004 and incorporated herein by reference: Method and program product for determining worst case currents in a digital integrated circuit through worst-case superposition of partial currents (Attorney Docket No. 200300123-1); Systems and methods that identify equivalent instantiation-specific configuration information for analysis tools (Attorney Docket No. 200308898-1); System and method for determining detail of analysis in a circuit design (Attorney Docket No. 200308899-1); System and method for indicating logic state combinations used during circuit design analysis (Attorney Docket No. 200308900-1); Systems and methods for re-using circuit design analysis results (Attorney Docket No. 200308901-1); System and method for balancing run-time and result accuracy in a circuit design analysis tool (Attorney Docket No. 200310170-1); System and method for determining control signal combinations for use during simulation of a stage of a circuit design (Attorney Docket No. 200310171-1); and System and method to limit analyzed current flow in a circuit design (Attorney Docket No. 200310173-1).

BACKGROUND

[0002] An electronic computer aided design ("E-CAD") package is utilized to construct a Very Large Scale Integration ("VLSI") circuit design. The VLSI circuit design consists of a netlist that identifies electronic design elements (e.g., capacitors, transistors, resistors, etc.) and their interconnectivity (e.g., signal nets) within the VLSI circuit design. The VLSI circuit design is constructed from hierarchical cells (also known as design blocks) that provide specific functionality to the VLSI circuit design. Cells may be constructed from electronic design elements, nets and other cells, and may be re-used one or more times. Each use or instantiation of a cell in the VLSI circuit design is called an "instance."

[0003] A signal net is a single electrical path in a circuit design that has the same electrical characteristics at all of its points. Any collection of wires that carries the same signal between design elements is a signal net. If the design elements allow the signal to pass through unaltered (as in the case of a terminal), then the signal net continues on subsequently connected wires. If, however, the design element modifies the signal (as in the case of a transistor or logic gate), then the signal net terminates at that design element and a new signal net begins on the other side.

[0004] A signal net may be divided into signal net 'pieces', each of which is part of a Highest Level Signal Name ("HLSN"). A HLSN is the unique signal name that identifies a collection of signal nets or 'hierarchical signal net pieces', which are the small pieces of intermediate wire (signal nets) in each hierarchical design block of a circuit design.

[0005] A significant characteristic of VLSI and other types of circuit design is a reliance on hierarchical description. A primary reason for using hierarchical description is to hide the vast amount of detail in a design. By reducing the distracting detail to a single object that is lower in the hierarchy, one can greatly simplify many E-CAD operations. For example, simulation, verification, design-rule checking, and layout constraints can all benefit from hierarchical representation, which makes them more computationally tractable. Since many circuit designs are too complicated to be easily considered in their totality, a complete circuit design is often viewed as a collection of design element aggregates that are further divided into sub-aggregates in a recursive and hierarchical manner. In VLSI circuit design, these aggregates are commonly referred to as design blocks or cells, as noted above. Each cell also typically has one or more 'ports'; each port provides a connection point between a signal net within the cell and a signal net external to the cell.

[0006] A design engineer uses the E-CAD tool to analyze the VLSI circuit design during development. The E-CAD tool typically selects a 'stage' within the circuit design for analysis. The stage is contained within one cell that may be instantiated one or more times in the VLSI circuit design; the VLSI circuit design, therefore, may contain more than one identical stage. Each stage in the circuit design has instantiation-specific configuration information that is used by the E-CAD tool during analysis of the stage. The instantiation-specific configuration information may

have several different sources, including input/output from other analysis tools and user input.

[0007] The E-CAD tool may perform several types of analyses (using different analysis tools, for example) on the VLSI circuit design, with each analysis utilizing particular configuration information. Each analysis tool that accesses configuration information typically parses textual configuration information contained in configuration files. These configuration files may therefore be parsed several times during analysis of the many stages of a VLSI circuit design.

[0008] If the VLSI circuit design contains billions of design elements and has many stages, the analyses can take hours or even days of processing time to complete, resulting in lost productivity. Continuous lost productivity due to lengthy engineering development slows technology advancement and can result in significant costs, as well as lost business.

SUMMARY OF THE INVENTION

[0009] In one embodiment, a method processes configuration information. One or more configuration elements are identified from one or more configuration commands. The configuration elements are associated with design elements of an electronic circuit design. Each configuration element is retrieved for at least one design element.

[0010] In another embodiment, a system processes configuration information, including: means for identifying at least one configuration element from at least one configuration command; means for associating the configuration element with one or more design elements of an electronic circuit design; means for retrieving each configuration element associated with at least one design element.

[0011] In another embodiment, a software product has instructions, stored on computer-readable media, wherein the instructions, when executed by a computer, perform steps for processing configuration information, including: instructions for identifying one or more configuration elements from one or more configuration commands; instructions for associating the configuration elements with design elements of an electronic circuit design; instructions for retrieving each configuration element for at least one design element.

BRIEF DESCRIPTION OF THE FIGURES

[0012] FIG. 1 illustrates one system embodiment for processing configuration information.

[0013] FIG. 2 is a schematic diagram of one cell with inverter functionality.

[0014] FIG. 3 is a block diagram illustrating one cell that twice instantiates the cell of FIG. 2.

[0015] FIG. 4 is a flowchart illustrating one embodiment of a method for processing configuration information.

[0016] FIG. 5 is a flowchart illustrating one embodiment of a method for processing configuration information.

DETAILED DESCRIPTION OF THE FIGURES

[0017] To analyze a circuit design (e.g., a very large scale integration (VLSI) circuit design), electronic computer aided design (E-CAD) tools often utilize vast amounts of configuration information. The speed at which the E-CAD tool accesses the configuration information thus impacts analysis tool performance. Certain systems and methods now described improve this access speed by making configuration information available on a net-by-net basis, and by enabling other E-CAD tools access to the configuration information.

[0018] FIG. 1 illustrates one system 100 that processes configuration information, such as to expedite CAD tool access to the configuration information. System 100 has a computer 102 with computer memory 104, a processor 106, a storage unit 108 and a user interface 110. Storage unit 108 is, for example, a disk drive that stores programs and data of computer 102. Storage unit 108 is illustratively shown storing an E-CAD tool 114, a circuit design 116, configuration information 130, and database 160. E-CAD tool 114 has a configuration tool 122. Circuit design 116 is, for example, a VLSI circuit design created by E-CAD tool 114. Configuration information 130 is illustratively shown with exemplary configuration commands 132 and 134.

[0019] Processor 106 loads E-CAD tool 114 from storage unit 108 into computer memory 104 such that E-CAD tool 114 is executable by processor 106. E-

CAD tool 114 may in turn request that processor 106 load configuration tool 122 and configuration information 130 from storage unit 108 into computer memory 104. E-CAD tool 114, configuration tool 122 and configuration information 130 are thus shown in dashed outline in computer memory 104, for purposes of illustration. Once loaded into computer memory 104, E-CAD tool 114 processes configuration information 130 to generate database 160, which may be stored in storage unit 108, as shown, or in computer memory 104 as a matter of design choice. Configuration information 130 may be loaded into computer memory 104 by either E-CAD tool 114 or configuration tool 122.

[0020] Configuration tool 122 creates a hierarchical model 140 based upon circuit design 116 in computer memory 104. Hierarchical model 140 is illustratively shown with two design elements 142 and 146. Design element 142 has a configuration element 144, and design element 146 has a configuration element 148. Design elements 142 and 146 are, for example, signal nets within circuit design 116.

[0021] By way of illustrative operation, user interface 110 connects to a terminal 112 (e.g., a keyboard) external to computer 102. Through terminal 112 and user interface 110, the design engineer interacts with E-CAD tool 114 and configuration tool 122. In one example, the design engineer instructs E-CAD tool 114 to process configuration information 130 for circuit design 116 using configuration tool 122. Configuration tool 122 then processes configuration information 130 to match design elements (e.g., design elements 142, 146) within hierarchical model 140 with one or more configuration commands (e.g., configuration commands 132, 134). In one example, design element 142 is matched to configuration command 132 and configuration information pertaining to design element 142 is stored in configuration element 144. Similarly, design element 146 is matched to configuration command 134 and configuration information pertaining to design element 146 is stored in configuration element 148. Configuration tool 122 then traverses hierarchical model 140 to produce database 160.

[0022] Configuration commands 132 and 134 may for example define rise and fall times of signals on signals nets within circuit design 116, logic configuration commands that define input signal combinations within circuit design 116, activity factors of signals nets within circuit design 116, drive fight periods that define the

percentage of time that two connected drivers drive opposing signal polarities, crossover current scale factors that specify the amount of time driver outputs compete when changing state, capacitance adjustments on various nets, etc.

[0023] Configuration information 130 may include logic commands that define states and logic relationships for signals nets within circuit design 116. Exemplary logic commands are given below in Example Logic Commands.

Example Logic Commands

```
Command 1:        if0then1 A B
Command 2:        if1then0 A B
Command 3:        if0then0 A B
Command 4:        if1then1 A B C D
Command 5:        mutex0 A B C D
Command 6:        mutex1 A B C D
Command 7:        imutex0 A B C D
Command 8:        imutex1 A B C D
Command 9:        set_high A
Command 10:       set_low A
```

[0024] In Example Logic Commands, Command 1 specifies that if signal A is 0 then signal B is set to 1. Command 2 specifies that if signal A is 1 then signal B is set to 0. Command 3 specifies that if signal A is zero then signal B is set to 0. Command 4 specifies that if signal A is 1 then signals B, C and D are set to 1. Command 5 specifies that exactly one of signals A, B, C and D is set to 0 at any one time. Command 6 specifies that exactly one of signals A, B, C and D is set to 1 at any one time. Command 7 specifies that no more than one of signals A, B, C and D is set to 1 at any time, although all signals A, B, C and D may be 0. Command 8 specifies that no more than one of signals A, B, C and D is set to 0 at any one time, although all signals A, B, C and D may be 1. Command 9 specifies that signal A is always set to 1. Command 10 specifies that signal A is always set to 0.

[0025] FIG. 2 shows one exemplary cell I0 that includes circuitry to provide inverter functionality. Cell I0 is, for example, suitable for use within circuit design 116, FIG. 1. Cell I0 has four ports 210, 212, 214 and 216 and, in this example, includes a p-type field-effect transistor ("FET") 204 and an n-type FET 206 connected

serially between port 214 and port 216. Port 214, labeled 'V', connects to power rail Voltage-Drain-Drain (VDD) and port 216, labeled 'G', connects to ground rail (GND). Cell I0 also has four signal nets 220, 222, 224 and 226: signal net 220 connects port 210 to gate terminals of FETs 204 and 206; signal net 222 connects a drain terminal of FET 204 and a source terminal of FET 206 (i.e., a common point between FETs 204 and 206) to output port 212; signal net 224 connects a source terminal of FET 204 to port 214; and signal net 226 connects a drain terminal of FET 206 to port 216. As shown in FIG. 3, cell I0 may be used within other cells to provide the inverter functionality, each use instantiating cell I0 within circuit design 116.

[0026] FIG. 3 illustrates one exemplary cell 300 with two instantiations I1, I2 of cell I0, FIG. 2. Cell 300 is, for example, suitable for use in circuit design 116, FIG. 1. Cell 300 has four ports 302, 304, 306 and 308. Port 302 is an input port that is connected to port 210(1) of cell instance I1 by signal net A. Port 212(1) of cell instance I1 is connected to port 210(2) of cell instance I2 by signal net B. Signal net C connects port 212(2) of cell instance I2 to output port 304. Port 308, labeled 'V', connects to power rail VDD and port 310, labeled 'G', connects to ground rail GND. Net 312 connects port 308 to ports 214(1) and 214(2); and net 314 connects port 310 to ports 216(1) and 216(2).

[0027] Using the example of FIG. 2 and FIG. 3, exemplary configuration commands are listed in Example Configuration Information below. Example Configuration Information is stored as configuration information 130, FIG. 1, for example.

Example Configuration Information

```
subckt I0 rise_time_receiver IN  2ps     # line 1
df * 0.0                                 # line 2
df C 0.01                                # line 3
rise_time_receiver B 1.3ps               # line 4
subckt I0 if1then0 IN OUT                # line 5
subckt I0 if0then1 IN OUT                # line 6
```

[0028] Referring to FIG. 2, Example Configuration Information line 1 specifies a rise time of 2.0 picoseconds for signals received on signal net IN of cell I0.

Accordingly, the configuration element associated with signal net IN for each instance of cell I0 (e.g., cell instances I1 and I2, FIG. 3) is updated to indicate a rise time of 2.0 picoseconds. Example Configuration Information line 2 uses a wildcard '*' to indicate that all signal nets within circuit design 116 have a drive fight factor of 0.0. Accordingly, all configuration elements associated with signal net design elements in hierarchical model 140 are updated to have a drive fight factor of 0.0. The drive fight factor represents the proportion of the clock period that a signal is in a drive fight condition. For example, a drive fight factor of 1.0 indicates that the signal is in the drive fight condition for the entire clock cycle; a drive fight factor of 0.5 indicates that the signal is in the drive fight condition for half the clock cycle. Example Configuration Information line 3 specifies a drive fight factor of 0.01 for signal net C, FIG. 3. Accordingly, only the configuration element associated with signal net C is updated with the drive fight factor 0.01, thereby overriding Example Configuration Information line 2 for signal net C. Example Configuration Information line 4 specifies a rise time of 1.3 picoseconds for signal net B. Accordingly, the configuration elements associated with signal net B is set to have a rise time of 1.3 picoseconds, thereby overriding Example Configuration Information line 1 for signal net B.

[0029] To illustrate exemplary processing of Example Configuration Information by configuration tool 122, consider signal net A as a first design element (e.g., design element 142) of hierarchical model 140. Configuration tool 122 matches the first design element to configuration commands (e.g., configuration command 132) within configuration information 130. For example, configuration tool 122 searches Example Configuration Information for configuration commands that apply to signal net A, and, in this example, determines: that line 1 applies since signal net A is the HLSN of signal net IN of cell instance I1; that line 2 applies since it uses a wildcard '*' that matches all signal nets; and that lines 5 and 6 apply since they identify signal net IN of cell I0, instantiated as cell I1 such that signal net A is the HLSN of signal net IN. Applicable configuration information of matched configuration commands (i.e., lines 1, 2, 5 and 6 of Example Configuration Commands, in this example) are stored in configuration elements (e.g., configuration element 144) within hierarchical model 140.

[0030] Once all design elements within hierarchical model 140 have been matched to configuration information 130, configuration tool 122 traverses hierarchical model 140 to generate database 160. Using the Example Configuration Information above, exemplary Database Record 1, Database Record 2, and Database Record 3 are now described.

[0031] Database Record 1, below, shows one exemplary database entry for signal net A, FIG. 3, resulting from processing Example Configuration Information, above, by configuration tool 122.

Database Record 1

```
# net A
Tr 2.0e-12          # from line 1
DF 0.0              # from line 2
I10 B               # from line 5
I01 B               # from line 6
```

[0032] In Database Record 1, the first configuration command specifies that the rise time for signal net A of 2.0 picoseconds, a value derived from line 1 of Example Configuration Information. The second configuration command in Database Record 1 specifies a drive fight factor of 0.0, derived from Example Configuration Information line 2. The third and fourth configuration commands in Database Record 1 are derived from lines 5 and 6 of Example Configuration Information, respectively, and specify that if signal net A is high, signal net B is low, and if signal net A is low, signal net B is high. Database Record 1 includes configuration information from configuration information 130 pertaining to signal net A of circuit design 116.

[0033] It should be apparent that Example Configuration Information and Database Record 1 represent one example of processing by configuration tool 122, and should not be construed as limiting. Other configuration commands may be included in both configuration information 130 and database 160 as a matter of design choice without departing from the scope herein.

[0034] Database Record 2, below, shows one exemplary database entry for signal net B, FIG. 3, resulting from processing of Example Configuration Information, above, by configuration tool 122.

Database Record 2

```
# net B
Tr 1.3e-12          # from line 4
DF 0.0              # from line 2
I10 C               # from line 5
I01 C               # from line 6
```

[0035] In Database Record 2, the first configuration command specifies that the rise time for signal net B of 1.3 picoseconds, a value derived from line 4 of Example Configuration Information. The second configuration command in Database Record 2 specifies a drive fight factor of 0.0, derived from Example Configuration Information line 2. The third and fourth configuration commands in Database Record 2 are derived from lines 5 and 6 of Example Configuration Information, respectively, and specify that if signal net B is high, signal net C is low, and if signal net B is low, signal net C is high. Database Record 2 includes configuration information from configuration information 130 pertaining to signal net B of circuit design 116.

[0036] Again it should be apparent that Example Configuration Information and Database Record 2 represent one example of processing by configuration tool 122, and should not be construed as limiting. Other configuration commands may be included in both configuration information 130 and database 160 as a matter of design choice without departing from the scope herein.

[0037] Database Record 3, below, shows one exemplary database entry for signal net C, FIG. 3, resulting from processing of Example Configuration Information, above, by configuration tool 122.

Database Record 3

```
# net C
DF 0.01             # from line 3
```

[0038] Database Record 3 includes one configuration command that specifies a drive fight factor of 0.01, derived from Example Configuration Information line 3. It should be apparent that Example Configuration Information and Database Record 3 represent one example of processing by configuration tool 122,

and should not be construed as limiting. Other configuration commands may be included in both configuration information 130 and database 160 as a matter of design choice without departing from the scope herein.

[0039] FIG. 4 is a flowchart illustrating one process 400 for processing configuration information. Process 400 is, for example, implemented by configuration tool 122, FIG. 1. In step 402, process 400 creates hierarchical model 140 in computer memory 104 from circuit design 116. In step 404, process 400 reads configuration information 130 into computer memory 104. In one example of step 404, process 400 reads one or more configuration files to generate configuration information 130 in computer memory 104. In step 406, process 400 matches one design element (e.g., design element 142) to identify zero or more configuration commands within configuration information 130 that provide configuration information pertinent to the one design element. For example, in step 406, process 400 matches configuration commands 132 to design element 142, thereby identifying configuration elements of configuration commands 132. In step 408, process 400 stores the identified configuration elements in hierarchical model 140. For example, in step 408, process 400 stores configuration information matched to design element 142 in configuration element 144, thereby associating configuration element 144 with design element 142. Steps 406 through 408 are repeated for each design element in hierarchical model 140, as indicated by section 410. In one example, process 400 sequentially selects each design element (e.g., design elements 142, 146) within hierarchical design 140, repeating steps 406 through 408 to store matched configuration information pertaining to each design element in associated data structures within hierarchical design 140.

[0040] Once all design elements have been matched to zero or more configuration commands (e.g., configuration elements 144, 148 are associated with design elements 142, 146, respectively), process 400 continues with step 412. In step 412, process 400 traverses hierarchical model 140, storing configuration elements (e.g., configuration elements 144, 148) of hierarchical model 140 in database 160 such that the information is associated with signal nets of circuit design 116. Database 160 thus allows one or more analysis tools to retrieve configuration elements associated with any selected design elements (e.g., a signal net) of circuit design 116 when analyzing or simulating circuit design 116.

[0041] FIG. 5 is a flowchart illustrating one process 500 for processing configuration information. Process 500 is, for example, implemented in configuration tool 122, FIG. 1. In step 502, process 500 identifies one or more configuration elements from one or more configuration commands. In step 504, process 500 associates the configuration elements with design elements of an electronic circuit design. In step 506, process 500 retrieves each configuration element for at least one design element.

[0042] As appreciated, hierarchical model 140 is more compact than configuration information 130; large configuration files can therefore be processed and stored in computer memory 104 allowing fast access to configuration elements 144, 148. Regular expression matching and parsing of configuration information 130 is consolidated and need only be performed once, evaluating configuration information 130 prior to use; errors and omissions of configuration information 130 may therefore be detected prior to initiation of one or more lengthy analyses. Since hierarchical model 140 is structured similarly to circuit design 116, configuration elements 144, 148 for a current analysis may be selectively and quickly retrieved from hierarchical model 140. Further, configuration elements 144, 148 are easily transformed at a global level, and additional configuration information may be derived and stored within hierarchical model 140 (requiring the derivation to be done only once).

[0043] Changes may be made in the above methods and systems without departing from the scope hereof. It should thus be noted that the matter contained in the above description or shown in the accompanying drawings should be interpreted as illustrative and not in a limiting sense. The following claims are intended to cover all generic and specific features described herein, as well as all statements of the scope of the present method and system, which, as a matter of language, might be said to fall there between.